

REMARKS/ARGUMENTS

Claims 1-22 are pending in the present application. Claims 2, 4-6, 9, 11-13, 16, and 18-20 were amended. Reconsideration of the claims is respectfully requested.

I. Examiner Interview Summary

Applicants thank Examiner Susan Rayyan for the courtesies extended to Applicants' representatives during the July 10, 2006 telephone interview. During the interviews, Applicants' representatives discussed the distinction between claim 1 features and *Loren* and *Bernstein* references. The Examiner agreed that the Applicants had identified several good distinctions between the claim and the references. No agreement as to the allowability of the claims was reached during the telephone interview.

II. 35 U.S.C. § 101

The Examiner has rejected claims 1-22 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter. This rejection is respectfully traversed.

The Examiner has rejected claims 1-22 stating:

A claim that requires one or more acts to be performed defines a process. However, not all processes are statutory under 35 U.S.C. 101. Schrader, 22 F.3d at 296, 30 USPQ2d at 1460. To be statutory, a claimed computer-related process must either: (A) result in a physical transformation outside the computer for which a practical application is either disclosed in the specification or would have been known to a skilled artisan, or (B) be limited to a practical application.

Claims 1, 8, 15, 22 in view of the above cited MPEP sections, are not statutory because they merely recite a number of computing steps without producing any tangible result and/or being limited to a practical application. The claim language recites "identifying data from a relational database and creating a temporary data set", "reading data from a relational database table", "preparing the data", "creating a database table for the relational database", and "placing the prepared data into the database table, wherein the prepared data is ready for analysis". The claim does not provide a concrete result as it does not store or display the results to the user.

Office action dated April 27, 2006, pp. 2-3.

Section 101 of Title 35 U.S.C. sets forth the subject matter that can be patented:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefore, subject to the conditions and requirements of this title.

"[N]o patent is available for a discovery, however useful, novel, and nonobvious, unless it falls within one of the express categories of patentable subject matter of 35 U.S.C. § 101." *Kewanee Oil Co. v. Bicron Corp.*, 416 U.S. 470, 483, 181 USPQ 673,679 (1974). The statutory categories of § 101 define eligible (patentable or statutory) subject matter, i.e., subject matter that can be patented. The listed statutory categories of invention are "process, machine, manufacture, or composition of matter."

Arguments advanced as to claim 1 are similarly applicable to claims 1-22. Claim 1 recites:

1. A method in a data preparation program for transferring data in a data processing system, the method comprising:
 - responsive to receiving user input, identifying data from a relational database and creating a temporary data set;
 - reading data from a relational database table in the relational database into the temporary data set created in response to the user input;
 - preparing the data read from the relational database table into the temporary data set to form prepared data for analysis;
 - creating a database table for the relational database; and
 - placing the prepared data into the database table, wherein the prepared data is ready for analysis.

The fact that data exists in a variety of storage systems is well known in the pertinent art. Data may exist in relational database, and non-relational database including flat-files, index files, delimited, hashed forms of data storage. Also well known in the pertinent art is the fact that data is seldom used from a solitary data source in isolation. The usefulness of data increases when data can be easily moved from one location to another, combined from sources of different characteristics, and analyzed together with other non co-residing data to yield meaningful and actionable results with practical applications.

Specific solutions have been created in order to transfer data from one type of database to another. Such solutions have improved efficiencies of data transfer, reliability of data transfer, or ease of data transfer. In fact, such solutions have been found to be patentable in the past. *See, Exporting and importing of data in object-relational databases, United States Patent No. 6,502, 098 (issued December 31, 2002); System and method for database synchronization, United States Patent No. 5,758,150 (issued, May 26, 1998).*

Claim 1 recites a method for transferring data into a data processing system. Particularly, the claim recites steps for transferring data from a relational database into a database of another kind. Steps of identifying and reading data from a relational database, preparing a temporary data set from the read data, and making the prepared data available for analysis in a different database, as recited in claim 1, clearly transfer data from a source database that makes the data usable in the destination database. While the practical applications of the transferred data depend upon the nature of the data being transferred, the act of transferring the data is useful in itself as has been found in the examples of issued patents cited above. Furthermore, the Examiner has cited, *Judy Loren, SAS Connection to DB2: Tools and Techniques*

(1996) (hereinafter, “*Loren*”), for obviousness rejection later in the office action. Some statements from *Loren* are applicable here because they validate the complexity of problems in the area of data transfer.

Loren states:

In the beginning there were Access Descriptors that allowed you to treat DB2 tables as if they were SAS datasets. Access Descriptors are simple to use, but take prohibitively long to run on large tables. Then came Pass-Through, so you could pass an entire SQL query directly to DB2 to execute. This allows DB2 to take advantage of its indexes, improving run time when all the information is in DB2. But what if you need to join a list of keys contained in a SAS dataset with a DB2 table (or tables)? This paper presents some suggestions based on real world experience using SAS as an access language for large DB2 databases.

To get SAS and DB2 to work together effectively, you need an array of tools and an understanding of when to use each of them.

Most of the challenges involved in using SAS with DB2 are performance related, primarily CPU consumption and clock time. Inefficiencies in these two areas can be insignificant with tables containing only hundreds of records. But when you try to use access descriptors on tables with millions of records--well, if you ever did, it's probably still running.

The first thing I should mention in discussing SAS and DB2 is access descriptors. These are like views that allow you to treat a DB2 table as if it were a SAS file. Now that I've mentioned them, forget them. When using large DB2 tables, the first thing you learn is DON'T USE ACCESS DESCRIPTORS. One reason is that when you use access descriptors, SAS doesn't deliver enough information to DB2 to allow it to make use of its indexes. As a result, everything is done as a table scan. Also, it seems that with Access Descriptors, there is some kind of record by record handshaking between SAS and DB2.

Loren, p. 498.

According to these statements from the Examiner’s cited reference, data transfer between DB2 and SAS can be complex and time consuming. Several tools mentioned in the Examiner’s cited reference are shown to have practical applicability in specific situations as the reference specifies. Undoubtedly, a method that facilitates data-transfer between databases, for example from DB2 to SAS, is therefore admittedly useful and produces tangible results. Such results are quantifiable at least in the time saved in data transfer and resources required for data transfer. Claim 1 recites such a method that is usable for transferring data between databases. Therefore, claim 1 is a claim to a useful method with practical application, and is patentable under 35 U.S.C. § 101.

Furthermore, claim 1 recites, “placing the prepared data into the database table,” which is a step reciting storage of data. The data is placed into a database table. Database tables exist on some physical medium, whether a hard-disk drive or integrated circuit memory chip, and therefore storing of data into a database table is necessarily a transformation of bits on some physical medium. The Examiner states,

“the claim does not provide a concrete result as it does not store or display the results to the user.” However, as described above, the claim in fact recites a storage in the last step – placing the prepared data into the database table – in direct contradiction to the Examiner’s assertion. For this additional reason, claim 1 is patentable under 35 U.S.C. § 101.

Independent claims 8, 15, and 22 contain features similar to those of independent claim 1, and are similarly patentable. Claims 2-7, 8-14, and 16-21 are also patentable at least by virtue of their dependence from one of these independent claims. Therefore, Applicants respectfully urge that claims 1-22 have practical application, recite patentable subject matter, and the rejection under 35 U.S.C. § 101 should be withdrawn.

III. 35 U.S.C. § 103, Obviousness

III.A. As to claims 1, 4-8, 11-15, and 18-22

The Examiner has rejected claims 1, 4-8, 11-15, 18-22 under 35 U.S.C. § 103 as being unpatentable over *Loren*, in view of *Bernstein* et al., Prefetching and caching persistent objects, United States Patent No. 6,728,726 B1 (issued April 27, 2004) (hereinafter, “*Bernstein*”). This rejection is respectfully traversed.

The Examiner has rejected claims 1, 8, 15, and 22 stating:

As per claims 1, 8, 15, 22 Loren teaches:
responsive to receiving user input, identifying data from a relational database (page 499, lines 9-11, query relational database and retrieve SAS dataset); preparing the data read from the relational database table . . . to form prepared data for analysis (page 499, lines 9-11, SAS dataset); creating a database table for the relational database; and placing the prepared data into the database table, wherein the prepared data is ready for analysis (page 499, lines 9-12).

Loren does not explicitly teach creating a temporary data set and reading data from a relational database table in the relational database into the temporary data set created in response to the user input. *Bernstein* does teach this limitation at col.7, lines 30-31. It would have been obvious to one of ordinary skill in the art at the time of the invention to provide storage for the results returned by a query (column 1, lines 65-67).

Office action dated April 27, 2006, pp. 3-4.

III.A.i. The Cited References Do Not Teach all of the Features of Claims 1, 8, 15, and 22

The Examiner has failed to state a *prima facie* obviousness rejection because the cited references used in proposed combination do not teach all of the features of claim 1 as believed by the examiner.

Claim 1 is representative of independent claims 8, 15, and 22 for the purpose of this rejection, and is quoted again for convenience:

1. A method in a data preparation program for transferring data in a data processing system, the method comprising:

responsive to receiving user input, identifying data from a relational database and creating a temporary data set;
reading data from a relational database table in the relational database into the temporary data set created in response to the user input;
preparing the data read from the relational database table into the temporary data set to form prepared data for analysis;
creating a database table for the relational database; and
placing the prepared data into the database table, wherein the prepared data is ready for analysis.

Contrary to the Examiner's assertion, *Loren* does not teach or even suggest the steps of "placing the prepared data into the database table, wherein the prepared data is ready for analysis" and "creating a database table for the relational database" as found in claim 1.

A *prima facie* case of obviousness is established when the teachings of the prior art itself suggest the claimed subject matter to a person of ordinary skill in the art. *In re Bell*, 991 F.2d 781, 783, 26 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1993). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). In the case at hand, not all of the features of the claimed invention have been considered and the teachings of the references themselves do not suggest the claimed subject matter to a person of ordinary skill in the art.

The Examiner has cited following sections from *Loren* as teaching, "placing the prepared data into the database table, wherein the prepared data is ready for analysis" feature of claim 1:

If all the data you need are in DB2 tables and you just want to use SAS to manipulate it, you have a friend in Pass Through access.

Loren, p. 499, ll. 9-11.

Examiner's citation cannot be treated in isolation for determining what is being taught by the cited section. A more complete section including the section cited by the Examiner is quoted below that provides the necessary information for better understanding of *Loren*'s teaching:

If all the data you need are in DB2 tables and you just want to use SAS to manipulate it, you have a friend in Pass Through access. Pass Through is a feature of SAS PROC SQL introduced in 6.07 that allows you to construct a DB2 SQL query which will be executed by DB2, with the results returned to SAS as either a SAS dataset if you used CREATE TABLE or a PROC PRINT if you didn't.

Pass Through access looks like this:

```
PROC SQL;
CONNECT TO DB2 (SSID=Q1);
%PUT &SQLXMSG;
CREATE TABLE OUT.OCT AS
SELECT *
```

```
FROM CONNECTION TO DB2
(SELECT *
FROM DSS.OA0041T_ORDER
WHERE ORD_TYPE_IND = 'M'
);
%PUT &SQLXMSG;
DISCONNECT FROM DB2;
%PUT &SQLXMSG;
```

You issue a CONNECT TO DB2 statement, giving the subsystem id in parentheses.

Loren, p. 499, first column.

According to the section quoted above, *Loren* teaches using a pass through access method when SAS, which is a flat-file database, needs to perform analysis of data that resides in DB2 relational database. *Loren* expressly states that the pass-through access feature of SAS constructs a query that is executed in DB2, that is, in the relational database, with the relational database writing the return code back to SAS. This teaching is in stark contrast to, “placing the prepared data into the database table, wherein the prepared data is ready for analysis” as recited in claim 1.

Claim 1 also recites, “creating a database table for the relational database.” These features make clear that the analysis is performed on the data, which has been placed into the database table, which is created in the database other than the relational database that is the source of the data. The cited reference teaches analysis in the relational database, and the claim, in contrast, recites analysis in a database other than relational database.

Loren further states:

The key to using Pass Through access is to remember that the piece inside the parentheses is not parsed by SAS but handed off as is to DB2 for execution.

Therefore, all the table names and field names inside the parentheses are full DB2 names.

Loren, p. 499, first column.

Through this statement, *Loren* further affirms that the analysis proceeds in the relational database, and not in the SAS database. This statement is contrary to the Examiner’s assertion of *Loren*’s teachings for the rejection, and in fact provides support for Applicants’ argument above. *Loren* teaches analysis of relational data in the relational database, whereas claim 1 recites analysis of data in the database other than the relational database. *Loren* and claim 1, therefore, pertain to different methods of data analysis and claim 1 features are not taught or suggested by *Loren*.

Elsewhere in a different section of *Loren*’s paper, *Loren* teaches a different method for transferring DB2 data into SAS, which also falls short of teaching all features of claim 1. The Examiner has not cited this section of *Loren*, quoted below, but Applicants further distinguish claim 1 from the

teaching present in the following section of *Loren*. *Loren*'s teaches using a Platinum tool to create flat files as follows :

If you decide to get your DB2 lookup table into SAS, you have a couple of options. A simple use of Pass Through can make the copy for you. This also allows you to join 2 or more tables to create one SAS dataset containing all the values you need from DB2. But if all you want is a subset of the fields or records from one DB2 table, you should consider a proprietary product called Platinum. Platinum goes at the VSAM files underlying DB2 and can create flat files from these at lightning speed. It allows you to select fields and records, but not join tables. You then have to read the flat files into SAS, but in a clocktime pinch it beats even pass through access for high volume reads of DB2.

Loren, p. 501, second column – p. 502, first column.

This section also fails to teach or suggest the features of “preparing the data read from the relational database table into the temporary data set to form prepared data for analysis” and “creating a database table for the relational database” as recited in claim 1. As a matter of express disclosure, *Loren*'s method of simply copying the data and using pass through access contains no disclosure of any preparation of data for analysis. Further, copying of data using pass through does not teach or suggest creation of a database table for relational database as recited in claim 1.

This section expressly states that Platinum uses VSAM (Virtual Storage Access Method technology by IBM™) files underlying DB2, not relational database tables themselves, and creates flat files from VSAM files, not database tables.

VSAM is an IBM licensed program; as an access method service, it provides fast storage and retrieval of data. Records are stored in control intervals; that is, in the block on the disk that VSAM uses to store data records and control information that describes the records. Records are ordered by key values in a key field or by when they were stored.

Source: IBM website for IBM TPF Information Center, Understanding Virtual Storage Access Method (VSAM) database support.

Relational database tables, on the other hand, are understood in the pertinent art as follows:

A relational database is a set of tables that have the following characteristics:
A table is made up of columns and rows.

A column is a set of values of the same datatype; a character column, for example, contains character strings and an integer column contains integers.

A row is a sequence of values such that the *n*th value of the row corresponds to the *n*th column of the table.

Each row is typically identified by a unique value known as its primary key. (It is possible, although not generally useful, to create a table without a primary key column.)

A base table is a table created with a CREATE TABLE statement. A base table persists in the database until it is removed with a DROP TABLE statement.

A result table is returned by a SELECT statement.

A temporary table is a table that is accessible only during the session in which it is created. A temporary table persists in the database only for the duration of that

session or until it is removed with a DROP TABLE statement.
Source: IBM website for IBM Red Brick Warehouse 6.3 Information Center, Overview of IBM Red Brick Warehouse.

For further details on VSAM technology, the Examiner may refer to the cited publication, “Understanding Virtual Storage Access Method (VSAM) database support,” which is included in an IDS for the Examiner’s convenience. For further details on relational database technology, the Examiner may refer to the cited publication, “Overview of IBM Red Brick Warehouse,” which is also included in the IDS.

These teachings in *Loren* are contrary to the features, “reading data from a relational database table in the relational database into the temporary data set,” as recited in claim 1. The methodology for reading VSAM files is different from methodology for reading a relational database table as is evident from the above quotations. One cannot teach the other primarily due to the difference in basic underlying technological concepts. Therefore, the second teaching in *Loren* is contrary to claim 1 features because reading VSAM files cannot teach reading data from a relational database table.

Additionally, *Loren*’s teaching in this section further fails to teach or suggest “preparing the data read from the relational database table into the temporary data set to form prepared data for analysis” because *Loren* states that Platinum does not join tables. Taking this statement in the context of *Loren*’s entire disclosure, *Loren* is teaching that Platinum simply provides flat file form of data to SAS leaving any preparation of data to SAS. Therefore, *Loren*’s teaching in this section also does not teach or suggest, “preparing the data read from the relational database table into the temporary data set to form prepared data for analysis” as recited in claim 1.

Because *Loren* teaches that Platinum provides flat file data as described above, *Loren*’s teaching is further contrary to yet another feature of claim 1. Claim 1 recites, “creating a database table for the relational database,” which expressly requires a database table, not flat file data to be created for the data from the relational database. *Loren*’s Platinum solution cannot teach creation of database table by teaching creating flat files. Therefore, *Loren*’s teaching in this section also does not teach or suggest, “creating a database table for the relational database” as recited in claim 1.

Therefore, neither the sections cited by the Examiner in *Loren*, nor the additional teachings of *Loren* identified by Applicants teach or suggest the steps of “preparing the data read from the relational database table into the temporary data set to form prepared data for analysis” and “creating a database table for the relational database” as recited in claim 1. Furthermore, *Loren*’s entire disclosure does not teach or suggest these features of claim 1. Consequently, Loren as a primary reference does not teach all the features of claim 1.

The Examiner cites *Bernstein* to teach the features of claim 1, “reading data from a relational database table in the relational database into the temporary data set,” and “creating a database table for the relational database,” deficient in *Loren*. The Examiner cites the following sections from *Bernstein* in support of the rejection:

In this embodiment, the storage cache is a temporary table that contains the results of queries.

In addition to being part of the state of an object, a structure can exist independently of other objects. For example, a structure may contain a group of objects that is the result returned by a data storage system when it executes a query.

Bernstein, col. 7, ll. 30-31; col. 1, ll. 65-67.

Bernstein teaches a method for pre-fetching data and holding the pre-fetched data till a request for such data is made. *Bernstein* provides support for this description as follows:

For the purposes of this application, prefetching means retrieving data for an object or related objects prior to an explicit request for access to the data by the application.

Bernstein, col. 2, ll. 59-61.

By definition, *Bernstein* teaches only making certain data available ahead of time, but not any preparation of such data. The “preparing” step in claim 1 is lacking in *Loren*’s entire disclosure as described above. *Bernstein* also does not teach or suggest preparation of data in the manner recited in claim 1. Specifically, *Bernstein* does not teach or suggest preparing the data read from the relational database table into the temporary data set to form prepared data for analysis.

In light of this deficiency in *Loren* as well as *Bernstein*, whether *Bernstein* teaches temporary data set is immaterial. Therefore, *Bernstein* fails to make up for the deficiencies in *Loren* with respect to claim 1. Therefore, *Loren* and *Bernstein*, considered separately or together, fail to teach or suggest all of the features of claim 1.

III.A.ii. The Examiner Has Not Stated a Proper Teaching, Suggestion or Motivation to Combine the References, and None Exists

In addition, a *prima facie* obviousness rejection against claim 1 has not been made because no proper teaching or suggestion to combine the references has been stated. A *prima facie* case of obviousness is established when the teachings of the prior art itself suggest the claimed subject matter to a person of ordinary skill in the art. *In re Bell*, 991 F.2d 781, 783, 26 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1993). A proper *prima facie* case of obviousness cannot be established by combining the teachings of the prior art absent some teaching, incentive, or suggestion supporting the combination. *In re Napier*, 55 F.3d 610, 613, 34 U.S.P.Q.2d 1782, 1784 (Fed. Cir. 1995); *In re Bond*, 910 F.2d 831, 834, 15 U.S.P.Q.2d

1566, 1568 (Fed. Cir. 1990). No such teaching or suggestion is present in the cited references and the Examiner has not pointed out any teaching or suggestion that is based on the prior art.

Instead, the Examiner has only stated a proposed advantage to combining the references. However, an advantage is not necessarily a teaching, suggestion, or motivation. To constitute a proper teaching, suggestion, or motivation, the Examiner must establish that one of ordinary skill would both recognize the advantage and have a reason to implement the advantage.

In the case at hand, the Examiner has not provided a sufficient reason why one of ordinary skill would recognize the proposed advantage or have a reason to implement it. The Examiner states that the motivation for doing so would have been “to provide storage for the results returned by a query.” However, the proposed motivation does not actually exist because *Loren*’s tools for SAS-DB2 connectivity already provide a mechanism for handling return values from DB2 and vitiate any putative need for storage for results of a query. For example, *Loren* states:

SQLXMSG contains the return code so you don’t need to write them both out.
Loren, p. 499, first column.

By disclosing this method for returning a result, *Loren* is not deficient in providing a mechanism for returning result values, and therefore removes any need for *Bernstein*’s method for returning results. For these reasons, the Examiner’s statement fails to provide a proper teaching, suggestion, or motivation to combine the references.

The references themselves do not teach or suggest the proposed advantage. In the present case, neither *Loren* nor *Bernstein* teaches a temporary storage space for results of queries in SAS-DB2 connectivity tools. Accordingly, the Examiner has failed to state a *prima facie* obviousness rejection against claim 1.

III.A.iii. *Loren* and *Bernstein* are Unrelated to Each Other as a Whole and One of Ordinary Skill in the Art Would Not Make the Suggested Combination to Reach the Invention in Claim 1

One of ordinary skill would not combine the cited references to reach claim 1 because the references are directed towards solving different problems when the cited references are considered as a whole. One of ordinary skill in the art considers the references as a whole, including the problem recognized and teachings provided by the references. It is necessary to consider the reality of the circumstances - in other words, common sense - in deciding in which fields a person of ordinary skill would reasonably be expected to look for a solution to the problem facing the inventor. *In re Oetiker*, 977 F.2d 1443 (Fed. Cir. 1992); *In re Wood*, 599 F.2d 1032, 1036, 202 U.S.P.Q. 171, 174 (CCPA 1979). The cited references do not address the same problems.

In the case at hand, *Loren* and *Bernstein* recognize different problems when considered in their entirety. *Loren* is directed towards tools and techniques for SAS-DB2 connectivity. For example, *Loren* provides that:

To get SAS and DB2 to work together effectively, you need an array of tools and an understanding of when to use each of them. First I will discuss several techniques; then I will give you guidelines to help decide when to use them.

Loren, p. 498, first column.

Loren identifies that various methods for integrating SAS and DB2 exist, but not all methods are equally applicable to all situations when designing integration solutions using the two databases. *Loren* is concerned with finding the most effective method for integrating SAS and DB2 databases under each of the design situations that *Loren* points out.

Bernstein is directed towards making access to object data more efficient by prefetching and caching object data. *Bernstein* provides:

While object-oriented databases and repositories provide a consistent mechanism for object persistence, the serial manner in which object-oriented applications access objects in persistent storage can cause undesirable performance degradation. In a typical data storage system, each access to a data item incurs a relatively high fixed overhead to interface with the persistent storage, and a relatively low incremental cost to actually retrieve the desired data item. This access cost is compounded by the fact that a large number of accesses are typically required to obtain moderate amounts of data related to a particular object or grouping of objects.

Therefore, there is a need in the art for a mechanism to increase the performance of object-oriented systems. The mechanism should allow applications to access objects using current methods, while taking advantage of common patterns of use to decrease the time required to access objects.

SUMMARY OF THE INVENTION

The above-mentioned shortcomings, disadvantages and problems are addressed by the present invention, which will be understood by reading and studying the following specification.

Bernstein, col. 2, ll. 33-55.

Bernstein identifies the problem of loss of efficiency when object data is serialized during access from a persistent storage. Bernstein points out that the overhead involved in each access to object data is the source of this problem and suggests a mechanism to overcome the problem and improve the object data transfer efficiency.

Each of these references recites a problem different from each other and further different from the problem addressed by the method of claim 1. Claim 1 recites a method for transferring data from a

relational database to another database where the data is transferred in a way that the data is ready for analysis at the destination database upon the completion of the transfer.

Furthermore, each reference provides complete solution for the problem each identifies. *Loren* states:

The first thing I should mention in discussing SAS and DB2 is access descriptors. These are like views that allow you to treat a DB2 table as if it were a SAS file. Now that I've mentioned them, forget them. When using large DB2 tables, the first thing you learn is DON'T USE ACCESS DESCRIPTORS. One reason is that when you use access descriptors, SAS doesn't deliver enough information to DB2 to allow it to make use of its indexes. As a result, everything is done as a table scan. Also, it seems that with Access Descriptors, there is some kind of record by record handshaking between SAS and DB2.

If all the data you need are in DB2 tables and you just want to use SAS to manipulate it, you have a friend in Pass Through access.

Next, we consider loading the SAS key values into DB2 and performing the merge there.

Writing to DB2

This option works only under the following conditions:

You have write access to the DB2 subsystem in which your lookup table resides, which is not necessarily a given.

The large table you want to extract data from is indexed on the key field you want to MERGE on.

If either of these conditions is not met, you shouldn't use this tool. Let's say for the moment, however, that you have met the conditions. Here is how we use this technique.

Write a flat file containing the key values.

Use a DB2 utility to load from the flat file to a DB2table.

Join the DB2 table of key values with other DB2 table(s) and write result to SAS file.

Loren, p. 498 second column, p. 499 first column, p. 500 first column.

Loren describes three distinct integration problems involving SAS and DB2 databases. *Loren* describes three complete solutions, one for each of the integration problems, and leaves no unresolved issues with respect to those problems and their suggested solutions, when the *Loren* disclosure is considered as a whole.

Bernstein also describes a complete solution for the problem *Bernstein* identifies. Just as *Bernstein*'s problem is distinct from *Loren*'s problem, *Bernstein*'s solution is also distinct from *Loren*'s solution when the two references are considered as a whole. *Bernstein* states:

The systems and methods presented implement various strategies to prefetch relevant data items when an application first accesses an initial data item included in the state of an object. For the purposes of this application, prefetching means retrieving data for an object or related objects prior to an explicit request

for access to the data by the application.

One such system for performing prefetches comprises a data storage system that provides persistent storage for object data comprising the state of the object. The data storage system uses an underlying physical storage system to actually store the data on a persistent storage unit. The physical storage system could be a hardware device, such as a disk, or a combination of hardware and software, such as a relational database system operating on a computer and disk. The data storage system includes software components that implement various strategies to prefetch data and store the data in a cache. The cache can be located in memory allocated to the data storage system, in memory allocated to an application using the data storage system to fetch object data, or in a physical storage system allocated to the data storage system. The data storage system or application to which the cache memory is allocated could be on a server system, a middle-tier system, or a client system.

In one method used to prefetch data, a data storage system provides access to an initial object whose state includes a structure that contains one or more other objects. The structure that contains those other objects is called the "structure context" of those other objects. The system creates a structure context description that "remembers" the objects in the structure and associates that structure context description with every object in the structure. When data for an attribute is fetched from one object in the structure (i.e. the object's structure context), data for the corresponding attribute is prefetched from the other objects in the structure, incurring a minimal incremental cost per item prefetched. The prefetched attribute data is held in a cache for later use. If the application later needs attribute data from an object in the structure, the system retrieves the attribute data from the cache if it is present, thereby avoiding the high fixed overhead cost of accessing the persistent storage for each attribute that can be successfully retrieved from the cache.

In an alternative method, a structure may be the result returned by the data storage system when it executes a query. The data storage system stores the structure, creates a structure context description that remembers the identifier of the stored structure, and associates that structure context description with every object in the structure. As before, when data for an attribute is fetched from one object in the structure, data for the corresponding attribute is prefetched from the other objects in the structure and held in a cache for later use.

Bernstein, col. 2, l. 56 – col. 3, l. 11; col. 2, ll. 30-39.

Bernstein summarizes two mechanisms to pre-fetch object data. *Bernstein* describes these mechanisms in order to reduce the data transfer overhead when object data is repeatedly accessed from persistent storage. *Bernstein* leaves no inadequacies in these mechanisms for addressing the identified problem when the *Bernstein* reference is considered as a whole.

Thus, the references address distinct problems that are unrelated to each other, and further unrelated to the invention of claim 1. The references also provide complete and distinct solutions of those distinct problems. Those solutions are further distinct from the recitation in claim 1. Therefore, one of

ordinary skill would have no reason to combine or otherwise modify the references to achieve the claimed invention. Thus, one of ordinary skill in the art would not combine these references as proposed by the Examiner. Accordingly, the Examiner has failed to state a *prima facie* obviousness rejection against claim 1.

III.A.iv. Summary of Why the Examiner Has Failed to State a *Prima Facie* Obviousness Rejection Against Claim 1.

In general, the Examiner appears to proceed from the false assumption that just because individual elements of a claimed invention can be found in two or more references, combining the references would automatically render the claimed invention obvious to one of ordinary skill. In fact, that vast bulk of patentable inventions are derived from combinations of elements that can be found in the prior art.

In the case at hand, the Examiner has failed to state a *prima facie* obviousness rejection for the following reasons: The proposed combination does not teach all of the features of claim 1; the Examiner has not stated a proper teaching, suggestion or motivation to combine the references, and none exists; and *Loren* and *Bernstein* would not be combined by one of ordinary skill in the art because they recognize and address different problems when considered as a whole. Therefore, the rejection against claim 1 under 35 U.S.C. § 103(a) has been overcome.

Independent claims 8, 15 and 22 contain features similar to those in claim 1. Arguments advanced against the obviousness rejection of claim 1 can similarly be applied against the obviousness rejection of claims 8, 15, and 22. Therefore, the rejection against claims 8, 15, and 22 under 35 U.S.C. § 103(a) has also been overcome. Claims 4-7, 11-14, and 18-21 are not rendered obvious by *Loren* in view of *Bernstein* at least by virtue of their dependence from one of these independent claims. Therefore, the rejection of claims 1, 4-8, 11-15, and 18-22 under 35 U.S.C. § 103(a) has been overcome.

III.B. As to claims 2-3, 9-10, 16-17

The Examiner has rejected claims 2-3, 9-10, and 16-17 under 35 U.S.C. § 103 as being unpatentable over *Loren*, in view of *Bernstein*, and further in view of *Waclawski et al., Meta data processing for converting performance data into a generic format*, United States Patent No. 6,128,628 (issued October 3, 2000) (hereinafter, “*Waclawski*”). This rejection is respectfully traversed.

The Examiner has rejected claims 2, 9, and 16 stating:

As per claims 2, 9, 16 Loren and Bernstein do not explicitly teach responsive to additional user input identifying data located in a plurality of database tables, reading data from the plurality of database tables into a plurality of data sets, merging the data from the plurality of data sets into a single data set, preparing the data to form additional prepared data and placing the additional prepared data in the single data set into the database table. Waclawski does teach these

limitations (column 5, lines 40-54). It would have been obvious to one of ordinary skill in the art at the time of the invention to combine the cited references to efficiently provide services such as capacity planning and other forecasting and diagnostic services (column 2, line 67- column 3, line 2).

Office action dated April 27, 2006, p. 5.

Claim 2 is representative of claims **2-3, 9-10, and 16-17** and in the currently amended form recites:

2. The method of claim 1 further comprising:
 - responsive to additional user input identifying data located in a plurality of database tables, reading data from the plurality of database tables into a plurality of data sets;
 - merging the data from the plurality of data sets into a single data set;
 - preparing the data from the plurality of data sets to form additional prepared data; and
 - placing the additional prepared data in the single data set into the database table.

III.B.i. The Proposed Combination Does Not Teach all of the Features of Claim 2

As shown above, *Loren* and *Bernstein* do not make obvious claim 1. Therefore, claim 2 is also patentable over *Loren* and *Bernstein* at least by virtue of the fact that claim 2 depends from claim 1.

Further, the Examiner has failed to state a *prima facie* obviousness rejection because the proposed combination does not teach all of the features of claim 2. Not all of the features of the claimed invention have been considered and the teachings of the references themselves do not suggest the claimed subject matter to a person of ordinary skill in the art.

Contrary to the Examiner's assertion, *Waclawski* does not teach, "identifying data located in a plurality of database tables, reading data from the plurality of database tables into a plurality of data sets." The Examiner has cited following sections from *Waclawski*:

The first program BLDSASDS produces one output SAS dataset **60** for multiple UDF input files **55**; the multiple UDF files **55** that result in a single output SAS dataset **60** are produced by the same type of collection agent (e.g., BMC Patrol), but represent different time segments and can come from multiple nodes (computers). In the preferred embodiment, BLDSASDS collects all UDF files for a single date from a single type of collection agent, and produces a single SAS dataset.

The second program, referred to herein as META PROCESSOR, takes the output SAS dataset **60** built by BLDSASDS and produces multiple Performance Data Tables **65**. In the preferred embodiment the Performance Data Tables **65** are in the form of an SAS dataset that is formatted for SAS IT Service Vision.

Waclawski, col. 5, ll. 40-54.

Examiner's interpretation of *Waclawski*'s UDF files as meaning data located in database tables as recited in claim 2 is erroneous. *Waclawski* defines UDF files as follows:

Most of the available collection agents may compile performance data into flat files known as Universal/Uniform Data Format (hereinafter UDF) files. *Waclawski*, col. 1, ll. 52-54.

Waclawski's UDF files are flat files, not database tables, and are compiled by collection agents, not data residing in some location such as database tables, as contrasted with recitations of claim 2. Consequently, *Waclawski* cannot teach the features of claim 2 that the Examiner alleges *Waclawski* teaches. In combination with the deficiencies in *Loren* and *Bernstein* described above with respect to claim 1, *Waclawski*'s entire disclosure further fails to teach other features of claim 2 that depend from claim 1. Therefore, *Loren*, *Bernstein*, and *Waclawski*, considered separately or together, fail to teach all features of claim 2.

III.B.ii. The Proposed Combination Changes the Principle of Operation of the Primary Reference

Because *Waclawski* deals with UDF data that is not the same as data residing in database tables as described above, *Waclawski*'s receiving and processing the UDF data follows none of the steps described in *Loren* for SAS-DB2 connectivity. UDF data in *Waclawski* is not described to support pass through access or other forms of access that *Loren* relies on for teaching SAS-DB2 connectivity tools and techniques. Thus, *Waclawski* necessarily changes *Loren*'s principle of operation if combined in the way the Examiner proposes. *In re Ratti* provides that changing the principle of operation of a device renders a claim non-obvious in view of the proposed combination. Therefore, claim 2 is non-obvious in view of the proposed combination.

III.B.iii. The Examiner Has Not Stated a Proper Teaching, Suggestion or Motivation to Combine the References, and None Exists

In addition, the Examiner has failed to state a *prima facie* obviousness rejection against claim 2 because the Examiner has not stated a proper teaching, suggestion, or motivation to combine the references. Instead, the Examiner has only stated a proposed advantage to combining the references. However, an advantage is not necessarily a teaching, suggestion, or motivation. To constitute a proper teaching, suggestion, or motivation, the Examiner must establish that one of ordinary skill would both recognize the advantage and have a reason to implement the advantage.

In the case at hand, the Examiner has not provided a sufficient reason why one of ordinary skill would recognize the proposed advantage or have a reason to implement it. The Examiner states that the motivation for doing so would have been "to efficiently provide services such as capacity planning and other forecasting and diagnostic services." However, the proposed motivation does not actually exist because neither *Loren* nor *Bernstein* expresses any desirability whatsoever for providing such services. Conversely, *Waclawski* suggests no need for connecting DB2 relational database to SAS as in *Loren*.

Waclawski also does not suggest any efficiency concerns in accessing object data as in *Bernstein*. No suggestion or motivation to combine *Loren*, *Bernstein*, and *Waclawski* in the manner posited by the Examiner actually exists in any of the references.

Anything is combinable with any other thing to some extent, and the possibilities from such combinations are not beyond the conjectures of a creative mind. However, unless the combined references themselves provide the motivation for such combination, the combination is improper for making obvious the features of a claimed invention. Because the Examiner's statement fails to provide a proper teaching, suggestion, or motivation to combine the references, the Examiner has failed to state a *prima facie* obviousness rejection against claim 2.

III.B.iv. Summary of Why the Examiner Has Failed to State a *Prima Facie* Obviousness Rejection Against Claim 2.

The Examiner has failed to state a *prima facie* obviousness rejection for the following reasons: The proposed combination does not teach all of the features of claim 2; combining the references would change the principle of operation of the primary reference; and the Examiner has not stated a proper teaching, suggestion or motivation to combine the references and none exists. Therefore, the rejection against claim 2 under 35 U.S.C. § 103(a) has been overcome.

Independent claims 3, 9-10, 16-17 contain features similar to those in claim 2. Arguments advanced against the obviousness rejection of claim 2 can similarly be applied against the obviousness rejection of claims 3, 9-10, and 16-17. Therefore, the rejection of claims 2-3, 9-10, and 16-17 under 35 U.S.C. § 103(a) has been overcome.

IV. Conclusion

It is respectfully urged that the subject application is patentable over *Loren, Bernstein* and *Waclawski*, and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: July 14, 2006

Respectfully submitted,



Rakesh Garg
Reg. No. 57,434
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Agent for Applicants